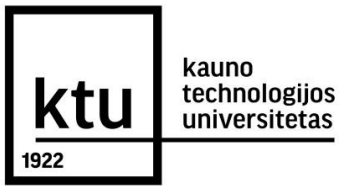


KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

T120B516 Objektinis programų projektavimas

Projekto aprašas

Studentai: Nedas Kasparavičius, IFF-7/13
Ugnius Mockus, IFF-7/3
Justinas Munius, IFF-7/5

KAUNAS 2020

Turinys

KAUNAS 2020

Projekto aprašymas

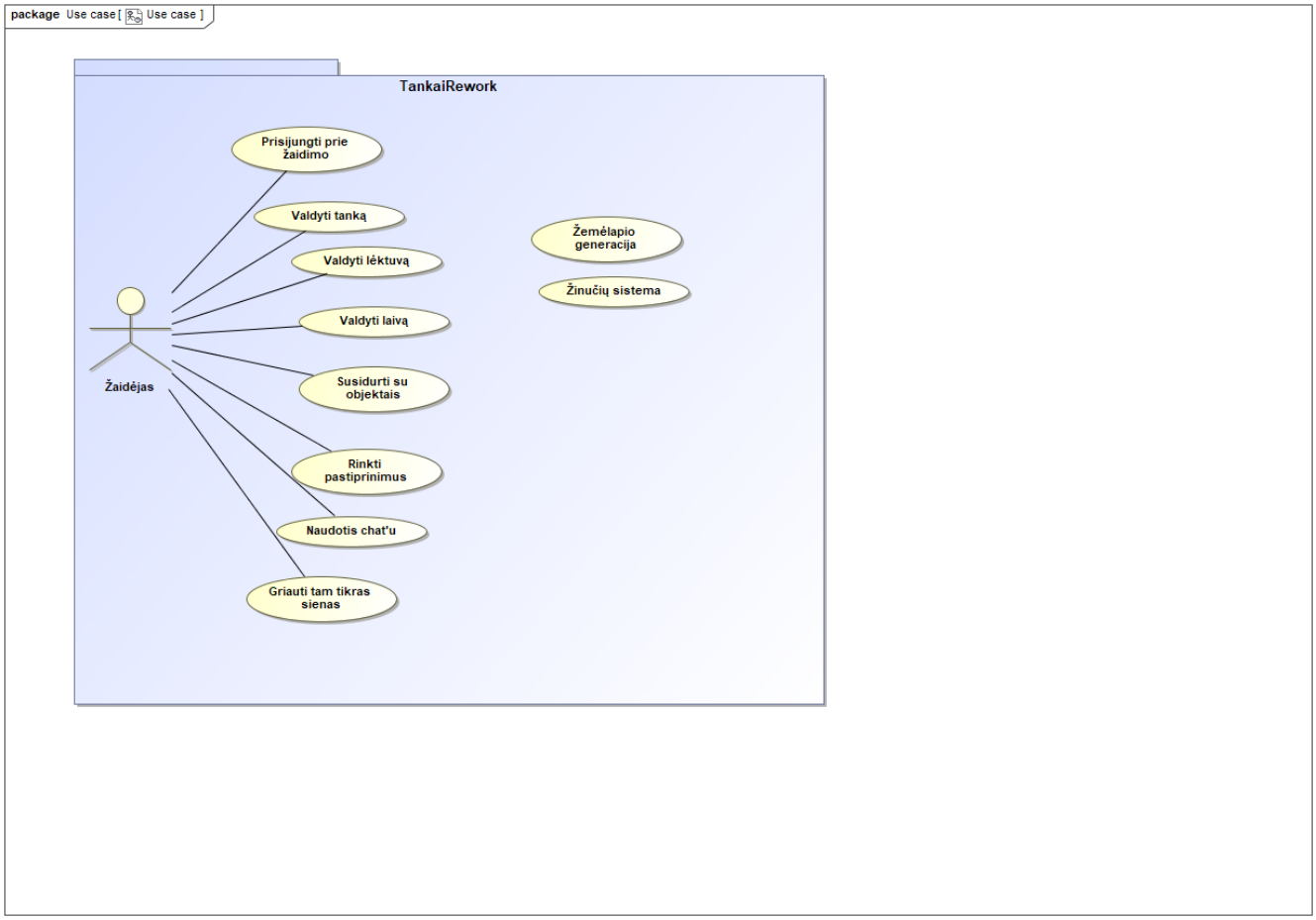
Kuriant „Tanks 2D” tipo žaidimą, kuris bus ėjimais grįstas, stengsimės išmokti taikyti projektavimo šablonus ir susipažinti su jų naudojimo ypatumais. Žaidime vartotojas valdys transportą (tanką, lėktuvą, laivą), kurių dėka stengsis įveikti visus kitus žaidėjus. Transporto priemonės galės šaudyti ir daryti žalą priešams, keliauti po žemėlapį. Žemėlapyje bus kelių skirtingų tipų langeliai, ant kurių galės arba negalės patekti veikėjas priklausomai nuo to, kokį transporto tipą jis bus pasirinkęs.

Naudojamos technologijos:

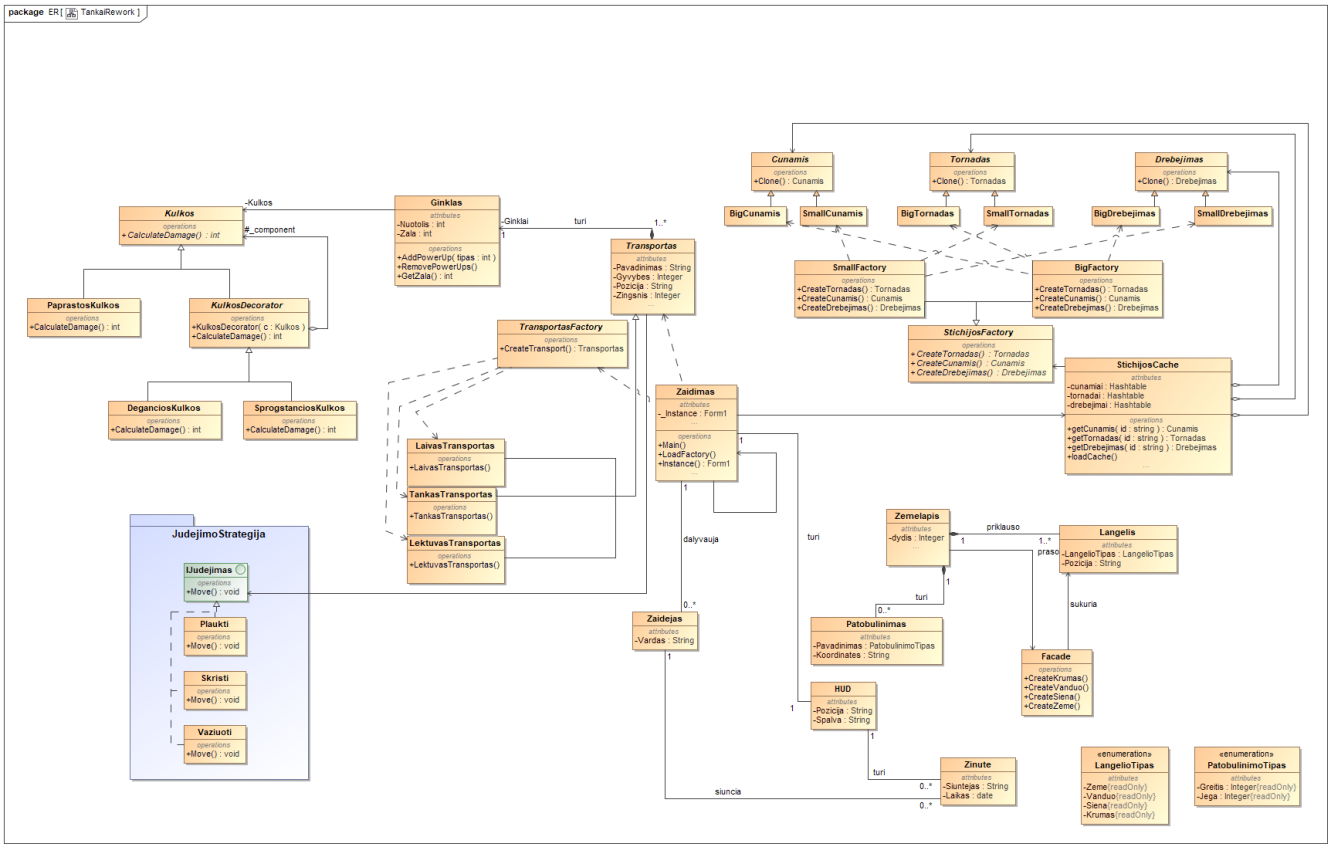
1. .Net Core
2. MVC
3. MongoDB
4. Rest API

KAUNAS 2020

Use Case Diagrama:



Klasių diagrama:



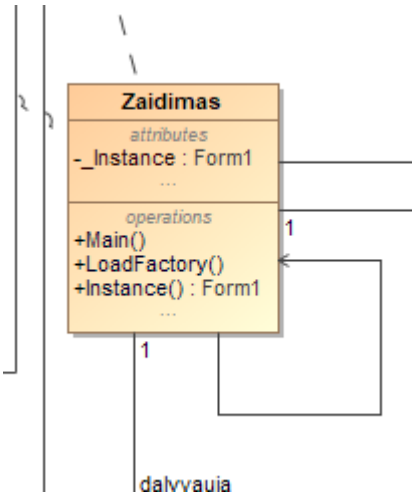
k

KAUNAS 2020

Projekto pirma dalis:

Singleton

Šį šabloną panaudojome, kad vienu metu būtų paleista tik viena programa

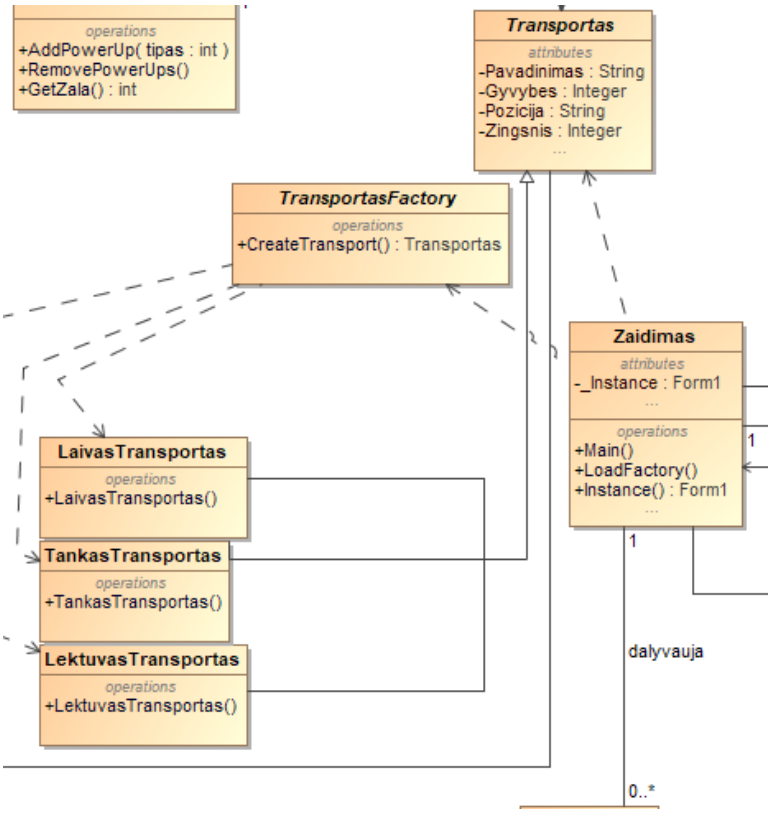


Programinis kodas:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace TanksRework
{
    static class Program
    {
        private static Form1 _instance;

        /// <summary>
        ///  The main entry point for the application.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.SetHighDpiMode(HighDpiMode.SystemAware);
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Form1 Instance = _instance ?? (_instance = new Form1());
            Application.Run(Instance);
        }
    }
}
```

Factory

Kadangi mūsų žaidėjai renkasi iš kelių transport priemonių (Laivas, Tankas, Lėktuvas), todėl naudojame factory šabloną kuris padės priskirti transporto priemones objektiškai



Programos kodas:

```
namespace Classes
{
    public class TransportasFactory
    {
        public Transportas CreateTransportas(int trClass, String
trName)
        {
            Random rnd = new Random();
            switch (trClass)
            {

                case 1:
                    {
                        return new LaivasTransportas(trName, 120, 8,
rnd.Next(1, 15), rnd.Next(1, 15) );
                    }
                case 2:
                    {
                        return new TankasTransportas(trName, 80, 12,
rnd.Next(1, 15), rnd.Next(1, 15) );
                    }
```

```
                case 3:
                    {
                        return new LektuvasTransportas(trName, 100,
10, rnd.Next(1, 15), rnd.Next(1, 15) );
                    }
                default:
                    return null;
            }
        }
    }
}
namespace Classes
{
    [JsonObject(MemberSerialization.OptIn)]
    public abstract class Transportas : Observer.Subject,
Observer.IObserver
    {
        [JsonProperty]
        private string _id { get; set; }
        [JsonProperty]
        private string name { get; set; }
        [JsonProperty]
        private int healthPoints { get; set; }
        [JsonProperty]
        private int damage { get; set; }
        [JsonProperty]
        public int positionx { get; set; }
        [JsonProperty]
        public int positiony { get; set; }
        [JsonProperty]
        public int type { get; set; }
        [JsonProperty]
        public IJudejimas _strategy { get; set; }


        public Transportas(String nam, int hp, int dmg, int posx, int
posy)
        {
            name = nam;
            healthPoints = hp;
            damage = dmg;
            positionx = posx;
            positiony = posy;
        }
        public Transportas(string id, String nam, int hp, int dmg, int
posx, int posy)
        {
            _id = id;
            name = nam;
            healthPoints = hp;
```

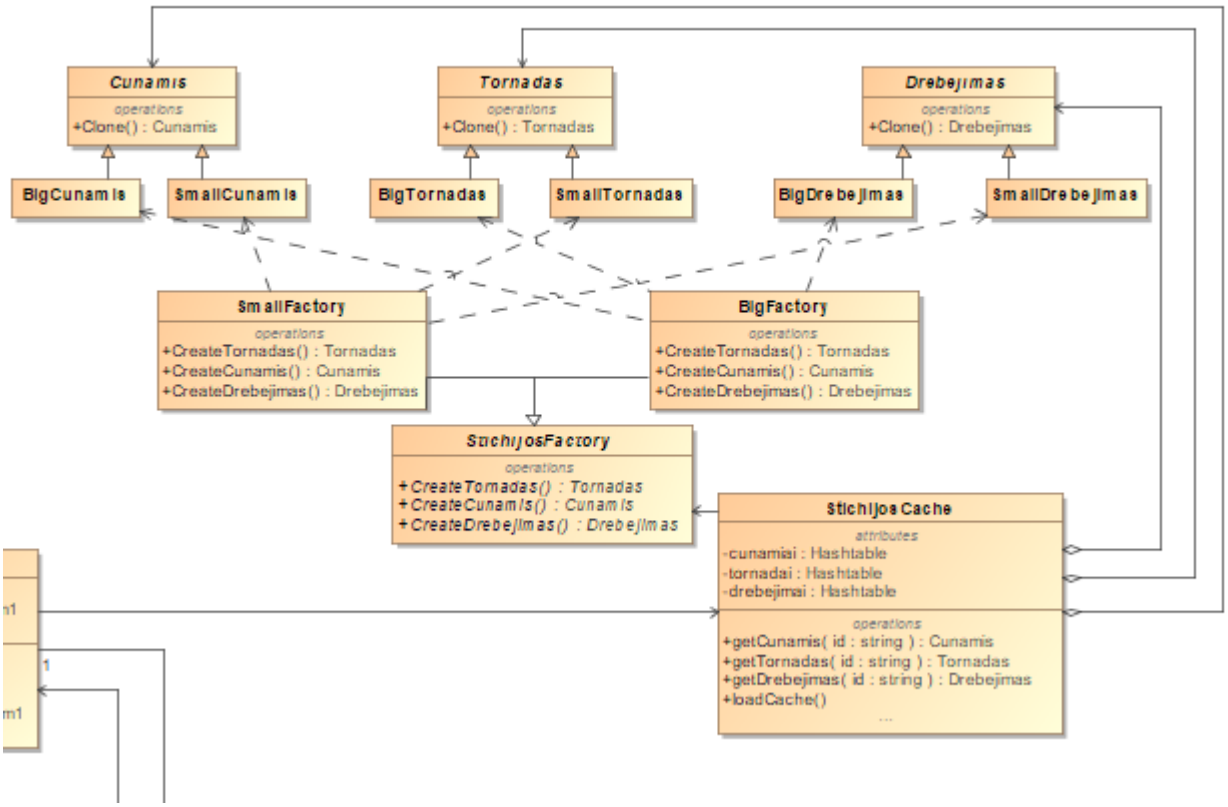
```
            damage = dmg;
            positionx = posx;
            positiony = posy;
        }
        public Transportas()
        {

        }

        public string getId()
        {
            return _id;
        }
        public void setId(string id)
        {
            _id = id;
        }

        void IObserver.atnaujinti(List<Transportas> updPriesai)
        {
            //Atnaujint info (pos = new pos)
            positionx = updPriesai.Find(p => p.getId() ==
_id).positionx;
            positiony = updPriesai.Find(p => p.getId() ==
_id).positiony;
            //if nera tokio id sarase, pridedam prie observeriu?
        }

        public void Move(int x, int y)
        {
            var positions = _strategy.Move(x, y, this.positionx,
this.positiony);
            this.positionx = positions.Item1;
            this.positiony = positions.Item2;
        }
    }
}
namespace Classes
{
    class TankasTransportas : Transportas
    {
        public TankasTransportas(String nam, int hp, int dmg, int
posx, int posy) : base(nam, hp, dmg, posx, posy)
        {
            type = 2;
            this._strategy = new Vaziuoti();
        }
        public TankasTransportas(string id, String nam, int hp, int
dmg, int posx, int posy) : base(id, nam, hp, dmg, posx, posy)
        {
```

```
            type = 2;
            this._strategy = new Vaziuoti();
        }
        public TankasTransportas()
        {
            type = 2;
            this._strategy = new Vaziuoti();
        }
    }
}

namespace Classes
{
    class LektuvasTransportas : Transportas
    {
        public LektuvasTransportas(String nam, int hp, int dmg, int
posx, int posy) : base(nam, hp, dmg, posx, posy)
        {
            type = 3;
            this._strategy = new Skristi();
        }
        public LektuvasTransportas(string id, String nam, int hp, int
dmg, int posx, int posy) : base(id, nam, hp, dmg, posx, posy)
        {
            type = 3;
            this._strategy = new Skristi();
        }
        public LektuvasTransportas()
        {
            type = 3;
            this._strategy = new Skristi();
        }
    }
}
namespace Classes
{
    class LaivasTransportas : Transportas
    {
        public LaivasTransportas(String nam, int hp, int dmg, int
posx, int posy) : base(nam, hp, dmg, posx, posy)
        {
            type = 1;
            this._strategy = new Plaukti();
        }
        public LaivasTransportas(string id, String nam, int hp, int
dmg, int posx, int posy) : base(id, nam, hp, dmg, posx, posy)
        {
            type = 1;
            this._strategy = new Plaukti();
        }
```

```
        public LaivasTransportas()
        {
            type = 1;
            this._strategy = new Plaukti();
        }
    }
}
```

AbstractFactor

Kadangi žaidime kursime stichijas, kurios gali būti arba didelės arba mažos, tam pasitelkiame abstract factory šabloną.



Programos kodas:

```
namespace TanksRework.Classes.AbstractFactory
{
    public abstract class StichijosFactory
    {
        public abstract Tornadas CreateTornadas();
        public abstract Cunamis CreateCunamis();
        public abstract Drebejimas CreateDrebejimas();
    }
}
namespace TanksRework.Classes.AbstractFactory
{
    public class SmallFactory : StichijosFactory
    {
        public override Tornadas CreateTornadas()
        {
            Random rnd = new Random();
            return new SmallTornadas(6, rnd.Next(1, 15), rnd.Next(1,
15));
```

```
        }
        public override Cunamis CreateCunamis()
        {
            Random rnd = new Random();
            return new SmallCunamis(6, rnd.Next(1, 15), rnd.Next(1,
15));
        }
        public override Drebejimas CreateDrebejimas()
        {
            Random rnd = new Random();
            return new SmallDrebejimas(6, rnd.Next(1, 15), rnd.Next(1,
15));
        }
    }
}
namespace TanksRework.Classes.AbstractFactory
{
    public class BigFactory : StichijosFactory
    {
        public override Tornadas CreateTornadas()
        {
            Random rnd = new Random();
            return new BigTornadas(6, rnd.Next(1, 15), rnd.Next(1,
15));
        }
        public override Cunamis CreateCunamis()
        {
            Random rnd = new Random();
            return new BigCunamis(6, rnd.Next(1, 15), rnd.Next(1,
15));
        }
        public override Drebejimas CreateDrebejimas()
        {
            Random rnd = new Random();
            return new BigDrebejimas(6, rnd.Next(1, 15), rnd.Next(1,
15));
        }
    }
}
namespace TanksRework.Classes.AbstractFactory
{
    [JsonObject(MemberSerialization.OptIn)]
    public abstract class Cunamis
    {
        [JsonProperty]
        private string _id { get; set; }
        [JsonProperty]
        private int damage { get; set; }
        [JsonProperty]
        public int positionx { get; set; }
```

```
        [JsonProperty]
        public int positiony { get; set; }
        [JsonProperty]
        public int type { get; set; }


        public Cunamis(int dmg, int posx, int posy)
        {
            damage = dmg;
            positionx = posx;
            positiony = posy;
        }
        public Cunamis(string id, int dmg, int posx, int posy)
        {
            _id = id;
            damage = dmg;
            positionx = posx;
            positiony = posy;
        }
        public Cunamis()
        {

        }
        public string getId()
        {
            return _id;
        }
        public void setId(string id)
        {
            _id = id;
        }
        public Cunamis Clone()
        {
            return (Cunamis)this.MemberwiseClone();
        }
    }
}
namespace TanksRework.Classes.AbstractFactory
{
    class SmallCunamis : Cunamis
    {
        public SmallCunamis(int dmg, int posx, int posy) : base(dmg,
posx, posy)
        {
            type = 2;
        }
        public SmallCunamis(string id, int dmg, int posx, int posy) :
base(id, dmg, posx, posy)
        {
            type = 2;
```

```
        }
        public SmallCunamis()
        {
            type = 2;
        }
    }
}
namespace TanksRework.Classes.AbstractFactory
{
    class BigCunamis : Cunamis
    {
        public BigCunamis(int dmg, int posx, int posy) : base(dmg,
posx, posy)
        {
            type = 1;
        }
        public BigCunamis(string id, int dmg, int posx, int posy) :
base(id, dmg, posx, posy)
        {
            type = 1;
        }
        public BigCunamis()
        {
            type = 1;
        }
    }
}
namespace TanksRework.Classes.AbstractFactory
{
    [JsonObject(MemberSerialization.OptIn)]
    public abstract class Tornadas
    {
        [JsonProperty]
        private string _id { get; set; }
        [JsonProperty]
        private int damage { get; set; }
        [JsonProperty]
        public int positionx { get; set; }
        [JsonProperty]
        public int positiony { get; set; }
        [JsonProperty]
        public int type { get; set; }


        public Tornadas(int dmg, int posx, int posy)
        {
            damage = dmg;
            positionx = posx;
            positiony = posy;
        }
```

```
        public Tornadas(string id, int dmg, int posx, int posy)
        {
            _id = id;
            damage = dmg;
            positionx = posx;
            positiony = posy;
        }
        public Tornadas()
        {

        }
        public string getId()
        {
            return _id;
        }
        public void setId(string id)
        {
            _id = id;
        }
        public Tornadas Clone()
        {
            return (Tornadas)this.MemberwiseClone();
        }
    }
}
namespace TanksRework.Classes.AbstractFactory
{
    class SmallTornadas : Tornadas
    {
        public SmallTornadas(int dmg, int posx, int posy) : base(dmg,
posx, posy)
        {
            type = 4;
        }
        public SmallTornadas(string id, int dmg, int posx, int posy) :
base(id, dmg, posx, posy)
        {
            type = 4;
        }
        public SmallTornadas()
        {
            type = 4;
        }
    }
}
namespace TanksRework.Classes.AbstractFactory
{
    class BigTornadas : Tornadas
    {
```

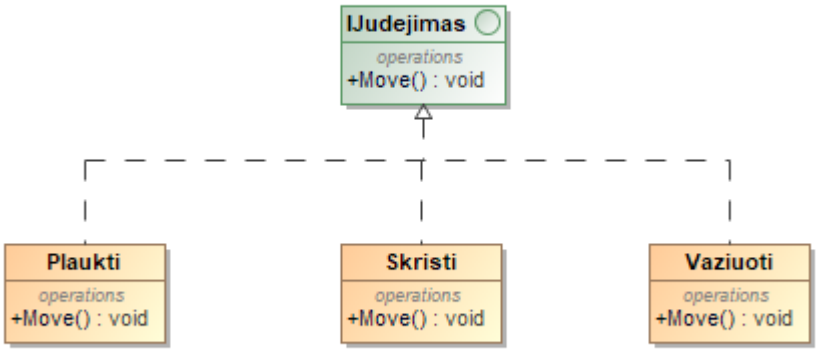
```
        public BigTornadas(int dmg, int posx, int posy) : base(dmg,
posx, posy)
        {
            type = 3;
        }
        public BigTornadas(string id, int dmg, int posx, int posy) :
base(id, dmg, posx, posy)
        {
            type = 3;
        }
        public BigTornadas()
        {
            type = 3;
        }
    }
}
namespace TanksRework.Classes.AbstractFactory
{
    [JsonObject(MemberSerialization.OptIn)]
    public abstract class Drebejimas
    {
        [JsonProperty]
        private string _id { get; set; }
        [JsonProperty]
        private int damage { get; set; }
        [JsonProperty]
        public int positionx { get; set; }
        [JsonProperty]
        public int positiony { get; set; }
        [JsonProperty]
        public int type { get; set; }


        public Drebejimas(int dmg, int posx, int posy)
        {
            damage = dmg;
            positionx = posx;
            positiony = posy;
        }
        public Drebejimas(string id, int dmg, int posx, int posy)
        {
            _id = id;
            damage = dmg;
            positionx = posx;
            positiony = posy;
        }
        public Drebejimas()
        {

        }
```

```
        public string getId()
        {
            return _id;
        }
        public void setId(string id)
        {
            _id = id;
        }
        public Drebejimas Clone()
        {
            return (Drebejimas)this.MemberwiseClone();
        }
    }
}
namespace TanksRework.Classes.AbstractFactory
{
    class SmallDrebejimas : Drebejimas
    {
        public SmallDrebejimas(int dmg, int posx, int posy) :
base(dmg, posx, posy)
        {
            type = 6;
        }
        public SmallDrebejimas(string id, int dmg, int posx, int
posy) : base(id, dmg, posx, posy)
        {
            type = 6;
        }
        public SmallDrebejimas()
        {
            type = 6;
        }
    }
}

namespace TanksRework.Classes.AbstractFactory
{
    class BigDrebejimas : Drebejimas
    {
        public BigDrebejimas(int dmg, int posx, int posy) : base(dmg,
posx, posy)
        {
            type = 5;
        }
        public BigDrebejimas(string id, int dmg, int posx, int posy) :
base(id, dmg, posx, posy)
        {
            type = 5;
        }
        public BigDrebejimas()
```

```
        {
            type = 5;
        }
    }
}
```

Strategy:

Šį šabloną panaudojome veikėjo vaikščiojimui, tam kad galėtume naudoti skirtingas funkcijas skirtingiems transporto tipams.



Programinis kodas:

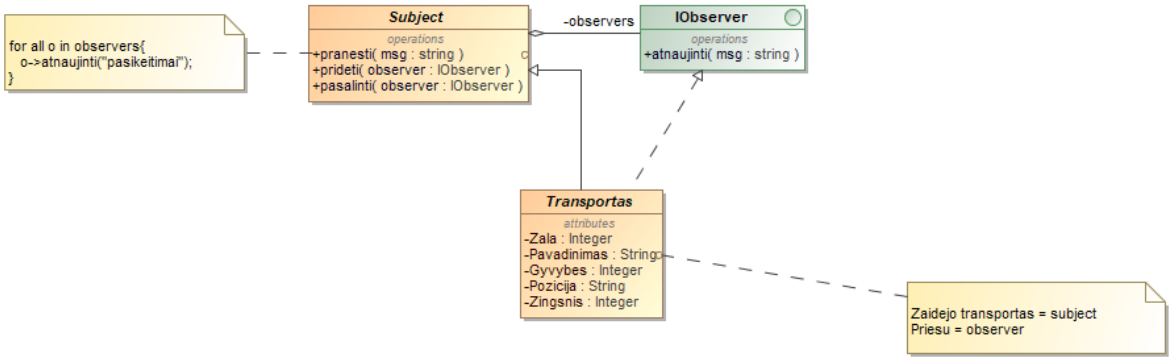
```
namespace


TanksRework.Classes.Strategy

{

    public interface IJudejimas

    {

        // x gali buti -1 arba 1, nuo to priklauso judejimas i kaire
ar desine

        // y asis yra apversta. +1 leidziasi juda zemyn, -1 juda
aukstyn

        public (int, int) Move(int x, int y, int posx, int posy);

    }

}
```

```
namespace TanksRework.Classes.Strategy
{
    class Plaukti : IJudejimas
    {
        public (int, int) Move(int x, int y, int posx, int posy)
        {
            return (posx += x, posy += y);
        }
    }
}

namespace TanksRework.Classes.Strategy
{
    class Skristi : IJudejimas
    {
        public (int, int) Move(int x, int y, int posx, int posy)
        {
            return (posx += x, posy += y);
        }
    }
}
namespace TanksRework.Classes.Strategy
{
    class Vaziuoti : IJudejimas
    {
        public (int, int) Move(int x, int y, int posx, int posy)
        {
            return (posx += x, posy += y);
        }
    }
```

}

Observer

Šablonas naudojamas norint informuoti priešų objektus apie pasikeitimus gavus atnaujintą informaciją iš serverio.



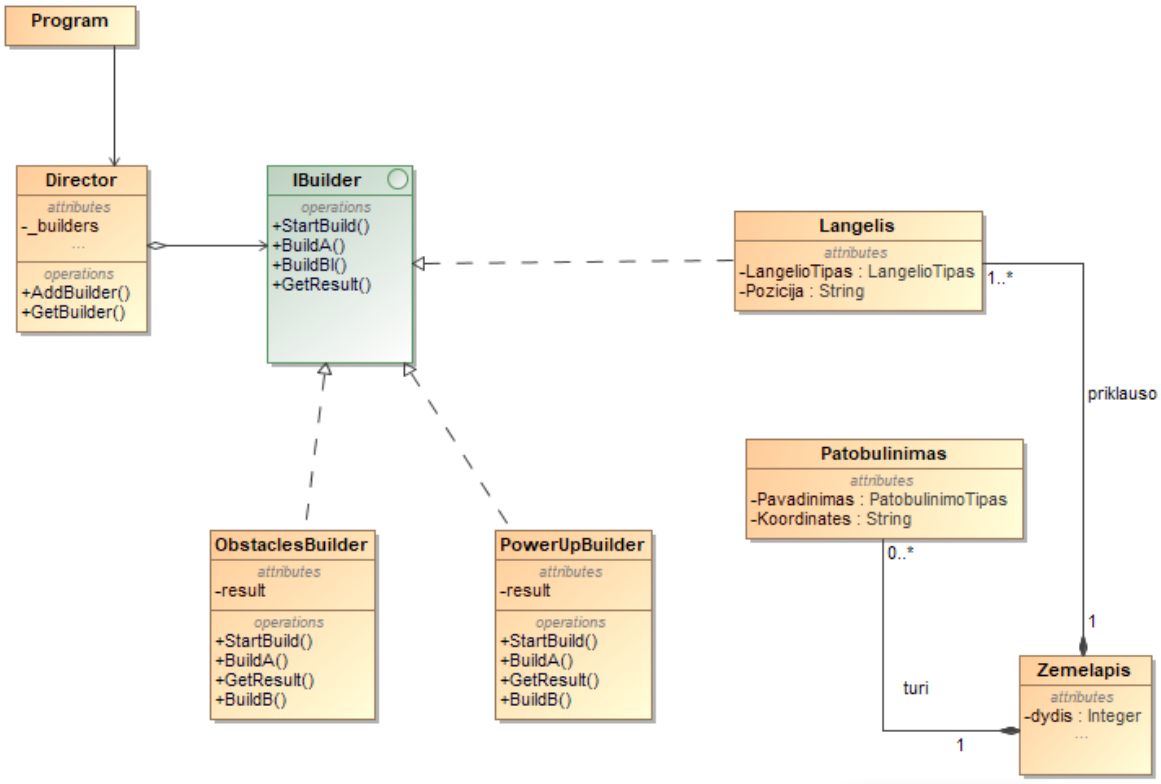
Programos kodas:

```
namespace Classes.Observer
{
    public interface IObserver
    {
        void atnaujinti(List<Transportas> updPriesai);
    }
}
namespace Classes.Observer
{
     public abstract class Subject
     {
          public List<IObserver> observers;

          public void pranesti(List<Transportas> priesai)//POSTina
savo pos ir serveri, mainais gauna priesu pos ir updeitina.
          {
               observers.ForEach(o =>
               {
                    o.atnaujinti(priesai);
               });
          }

          public void prideti(IObserver observer)
          {
               observers.Add(observer);
          }
```

```
          public void pasalinti(IObserver observer)
          {
               observers.Remove(observer);
          }

     }
}
public abstract class Transportas : Observer.Subject, Ob-
server.IObserver
    {
        void IObserver.atnaujinti(List<Transportas> updPriesai)
        {
            //Atnaujint info (pos = new pos)
            positionx = updPriesai.Find(p => p.getId() == _id).posi-
tionx;
            positiony = updPriesai.Find(p => p.getId() == _id).posi-
tiony;
            //if nera tokio id sarase, pridedam prie observeriu?
        }
    }
}
```

Builder

Šį šabloną naudojome serverio pusėje, tam kad būtų paprastą sukurti žaidimui žemėlapį, kuris bus sudarytas iš langelių, kuris kiekvienas priklausomai nuo tipo turės tam tikrą paveiksliuką.



Programinis kodas:

```
namespace TankaiServer.Classes.Builder
{
    class Director
    {
        public void Construct(IBuilder builder)
        {
            builder.StartBuild();
        }
    }
}
```

```
namespace TankaiServer.Classes.Builder
{
    interface IBuilder
    {
        void BuildA();

        void BuildB();
        void StartBuild();
        int[,] GetResult();
    }
}

namespace TankaiServer.Classes.Builder
{
    public class ObstacleBuilder : IBuilder
    {
        private int[,] map = new int[15, 15];
        Random rnd = new Random();
        public ObstacleBuilder(int[,] matrix)
        {
            //map = matrix;
        }
        public void BuildA()
        {
            for (int k = 0; k < map.GetLength(0); k++)
            {
                for (int l = 0; l < map.GetLength(1); l++)
                {
                    map[k, l] = 1;
                }
```

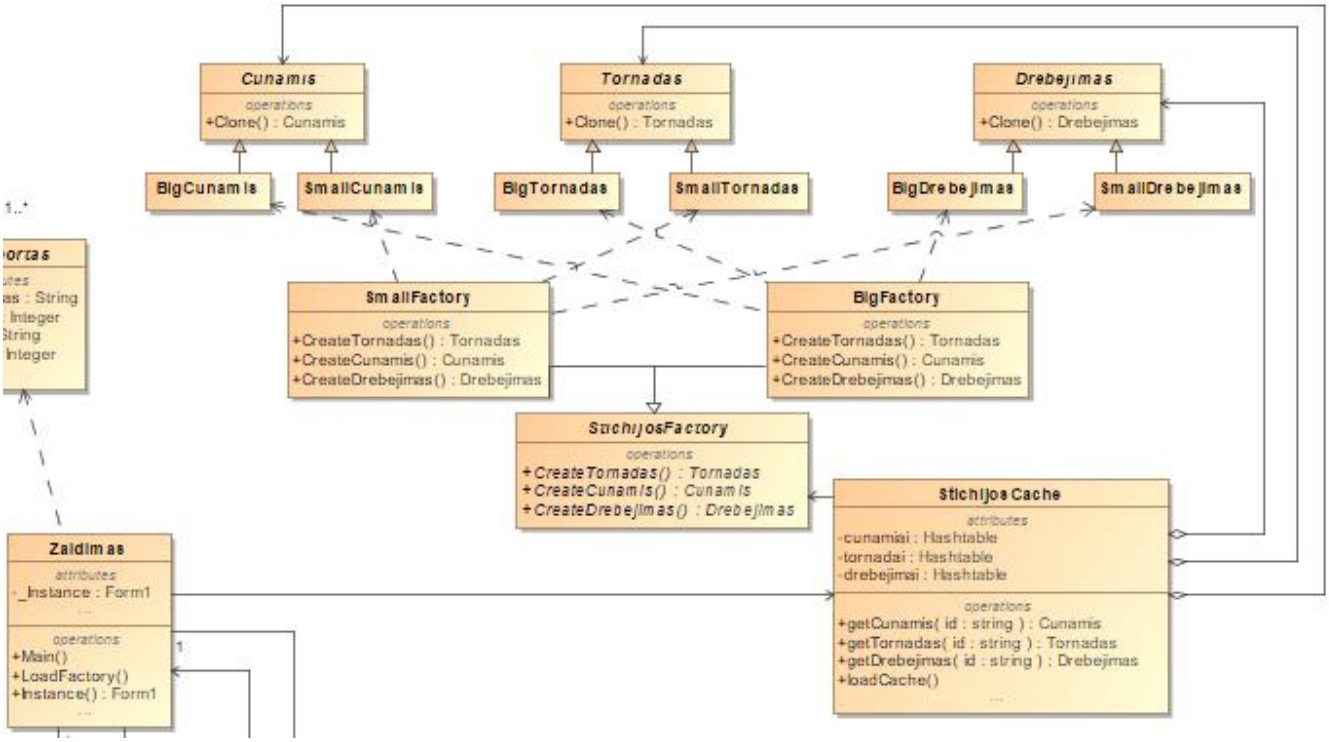
```
            }
            for (int k = map.GetLength(0) - 6; k < map.GetLength(0);
k++)
            {
                for (int l = 0; l < map.GetLength(1); l++)
                {
                    map[k, l] = 2;
                }
            }
        }

        public void BuildB()
        {
            int i = 0;
            float count = map.GetLength(0) * map.GetLength(1) / 5;
            while(i < count)
            {
                map[rnd.Next(0, map.GetLength(0)), rnd.Next(0,
map.GetLength(1))] = 3;

                i++;
            }

            i = 0;

            while (i < count / 2)
            {
                map[rnd.Next(0, map.GetLength(0)), rnd.Next(0,
map.GetLength(1))] = 4;

                i++;
```

```
            }
        }

        public int[,] GetResult()
        {
            return map;
        }

        public void StartBuild()
        {
            BuildA();
            BuildB();
        }
    }
}

namespace TankaiServer.Classes.Builder
{
    public class PowerUpBuilder : IBuilder
    {
        public int[,] map;
        Random rnd = new Random();

        public PowerUpBuilder(int[,] matrix)
        {
            map = matrix;
        }
        public void BuildA()
        {
            int i = 0;
```

```
            while(i < 3)
            {
                map[rnd.Next(0, 15), rnd.Next(0, 15)] = 5;
                i++;
            }
        }

        public void BuildB()
        {

        }

        public int[,] GetResult()
        {
            return map;
        }

        public void StartBuild()
        {
            BuildA();
            BuildB();
        }
    }
}
```

Prototype

Šis šablonas supaprastina kliento kodą norint sukurti naujas stichijas. Paleidus žaidimą užkraunamos visos stichijos, norint gauti konkrečią stichija nekuriamas naujas objektas, o paduodama jau sukurto objekto kopija



Programos kodas:

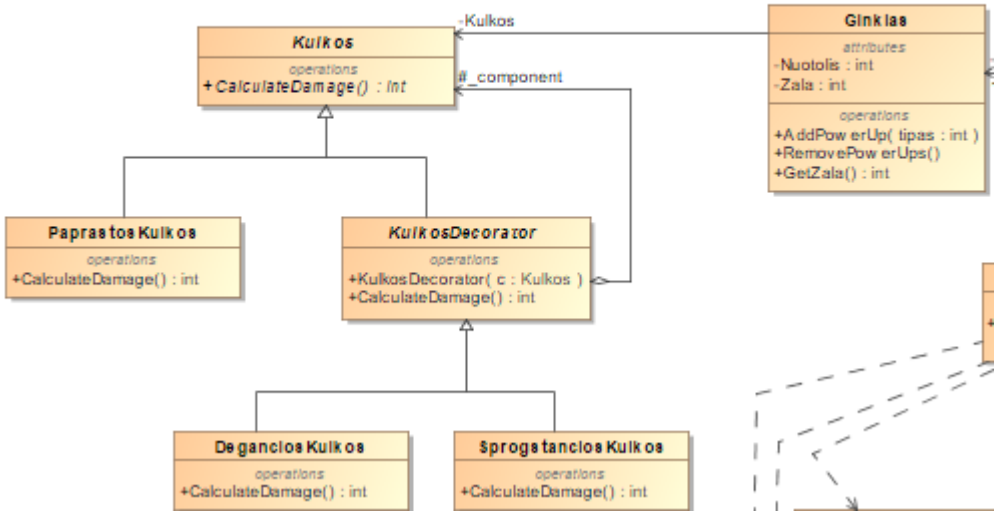
```
namespace TanksRework.Classes.AbstractFactory
{
    class StichijosCache
    {
        private Hashtable cunamiai;
        private Hashtable tornadai;
        private Hashtable drebejimai;

        public Cunamis GetCunamis(string id)
        {
            if (cunamiai.ContainsKey(id))
            {
                Cunamis cachedCunamis = (Cunamis)cunamiai[id];
                return (Cunamis)cachedCunamis.Clone();
            }
            else
            {
                return null;
            }
        }
```

```
        public Tornadas GetTornadas(string id)
        {
            if (tornadai.ContainsKey(id))
            {
                Tornadas cachedTornadas = (Tornadas)tornadai[id];
                return (Tornadas)cachedTornadas.Clone();
            }
            else
            {
                return null;
            }
        }
        public Drebejimas GetDrebejimas(string id)
        {
            if (drebejimai.ContainsKey(id))
            {
                Drebejimas cachedDrebejimas = (Drebejimas)drebe-
jimai[id];
                return (Drebejimas)cachedDrebejimas.Clone();
            }
            else
            {
                return null;
            }
        }
        public void loadCache()
        {
            BigFactory bigFactory = new BigFactory();
            SmallFactory smallFactory = new SmallFactory();

            cunamiai = new Hashtable();
            tornadai = new Hashtable();
            drebejimai = new Hashtable();

            Cunamis bCunamis = sukurtiCunami(bigFactory);
            bCunamis.setId("1");
            cunamiai.Add(bCunamis.getId(), bCunamis);

            Cunamis sCunamis = sukurtiCunami(smallFactory);
            sCunamis.setId("2");
            cunamiai.Add(sCunamis.getId(), sCunamis);

            Tornadas bTornadas = sukurtiTornada(bigFactory);
            bTornadas.setId("3");
            tornadai.Add(bTornadas.getId(), bTornadas);

            Tornadas sTornadas = sukurtiTornada(smallFactory);
            sTornadas.setId("4");
            tornadai.Add(sTornadas.getId(), sTornadas);
```

```
            Drebejimas bDrebejimas = sukurtiDrebejima(bigFactory);
            bDrebejimas.setId("5");
            drebejimai.Add(bDrebejimas.getId(), bDrebejimas);

            Drebejimas sDrebejimas = sukurtiDrebejima(smallFactory);
            sDrebejimas.setId("6");
            drebejimai.Add(sDrebejimas.getId(), sDrebejimas);
        }

        private Cunamis sukurtiCunami(StichijosFactory factory)
        {
            return factory.CreateCunamis();
        }
        private Tornadas sukurtiTornada(StichijosFactory factory)
        {
            return factory.CreateTornadas();
        }
        private Drebejimas sukurtiDrebejima(StichijosFactory factory)
        {
            return factory.CreateDrebejimas();
        }
    }
}
namespace TanksRework.Classes.AbstractFactory
{
    public abstract class Cunamis
        public Cunamis Clone()
        {
            return (Cunamis)this.MemberwiseClone();
        }
    }
}
```

Decorator

Šis šablonas naudojamas tam, kad suteikti papildomo funkcionalumo kulkom žaidėjui paėmus atitinkamą powerup'ą suteikiama papildoma savybė. Powerup'ai gali būti naudojami keli vienu metu.



Programos kodas:

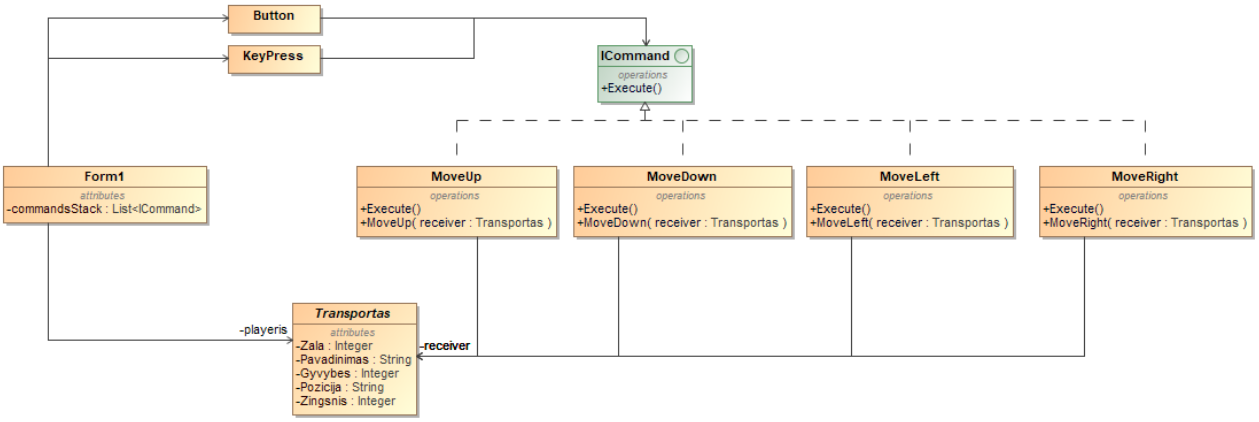
```
namespace TankaiRework.ER
{
     public class Ginklas
     {
          int Nuotolis;
          int Zala;
          Kulkos Kulkos;

          public Ginklas(int nuotolis, int zala)
        {
               Zala = zala;
               Nuotolis = nuotolis;
               Kulkos = new PaprastosKulkos();
        }

          public void AddPowerUp(int tipas) //1 - degancios kulkos, 2
- sprogstancios
        {
            switch (tipas)
            {
                    case 1:
                         Kulkos = new DeganciosKulkos(Kulkos);
                         break;
                    case 2:
```

```
                        Kulkos = new SprogstanciosKulkos(Kulkos);
                        break;
                default:
                    break;
            }
        }
          public void RemovePowerUps()
        {
              Kulkos = new PaprastosKulkos();
        }

          public int GetZala()
        {
              return Kulkos.CalculateDamage(Zala);
        }

    }

}
namespace TankaiRework.ER
{
    public abstract class Kulkos
    {
        public abstract int CalculateDamage( int zala );

    }

}
namespace TankaiRework.ER
{
    public class PaprastosKulkos : Kulkos
    {
        public override int CalculateDamage(int zala)
        {
            return zala;
        }

    }

}
namespace TankaiRework.ER
{
    public abstract class KulkosDecorator : Kulkos
    {
        protected Kulkos _component;

        public KulkosDecorator( Kulkos c )
        {
            this._component = c;
        }
```

```
          public override int CalculateDamage(int zala)
          {
               if (this._component != null)
               {
                    return this._component.CalculateDamage(zala);
               }
               else
               {
                    return 0;
               }
          }

     }

}
namespace TankaiRework.ER
{
     public class DeganciosKulkos : KulkosDecorator
     {
          public DeganciosKulkos(Kulkos c) : base(c)
        {
        }
          public override int CalculateDamage(int zala)
          {
               return base.CalculateDamage(zala) * 2;
          }
     }

}
namespace TankaiRework.ER
{
     public class SprogstanciosKulkos : KulkosDecorator
     {
          public SprogstanciosKulkos(Kulkos c) : base(c)
        {
        }
          public override int CalculateDamage(int zala)
          {
               return base.CalculateDamage(zala) * 5;
          }

     }

}
```

Command

Šablonas naudojamas norint pridėti papildomą sluoksnį ir atskirti kliento kodą nuo funkcijų.logikos. Tuos pačius metodus galimą kviesti keliais būdais: per gui, arba klaviatūros mygtukais (WASD). Funkcionalumas lengvai išplėčiamas.



Programos kodas:

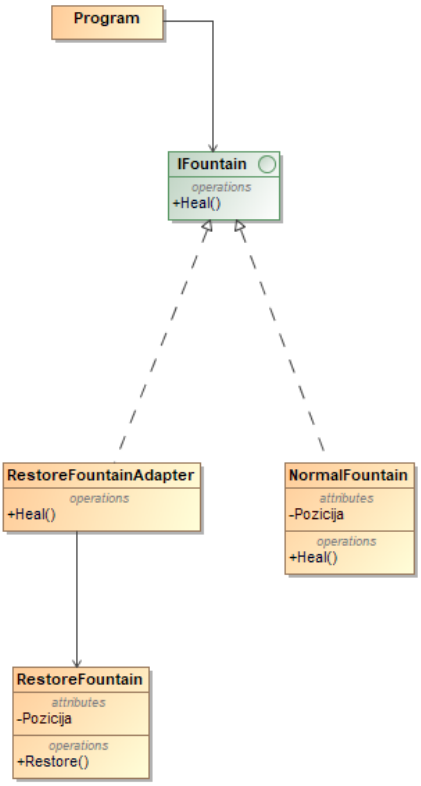
```
namespace TankaiRework.Commander
{
	public interface ICommand
	{
		void Execute(  );

	}

}
namespace TankaiRework.Commander
{
	public class MoveDown : ICommand
	{
		Transportas receiver;

		public void Execute(  )
		{
			this.receiver.Move(0, 1);
		}

		public MoveDown( Transportas receiver )
		{
			this.receiver = receiver;
		}

	}
```

```
}
namespace TankaiRework.Commander
{
     public class MoveLeft : ICommand
     {
          Transportas receiver;

          public void Execute(  )
          {
               this.receiver.Move(-1, 0);
          }

          public MoveLeft( Transportas receiver )
          {
               this.receiver = receiver;
          }

     }

}
namespace TankaiRework.Commander
{
     public class MoveRight : ICommand
     {
          Transportas receiver;

          public void Execute(  )
          {
               this.receiver.Move(1, 0);
          }

          public MoveRight( Transportas receiver )
          {
               this.receiver = receiver;
          }

     }

}
namespace TankaiRework.Commander
{
     public class MoveUp : ICommand
     {
          Transportas receiver;

          public void Execute(  )
          {
               this.receiver.Move(0, -1);
          }
```

```
        public MoveUp( Transportas receiver )
        {
            this.receiver = receiver;
        }

    }

}
```

Adapter

Šis šablonas naudojamas panašių klasių, tik skirting metodų suvienodinimui, kurie veikia panašia logika. Šiuo atveju Adapteris pritaikomas RestoreFountain klasei Adaptuoti iš Restore() metodo į Heal().



Programos kodas:

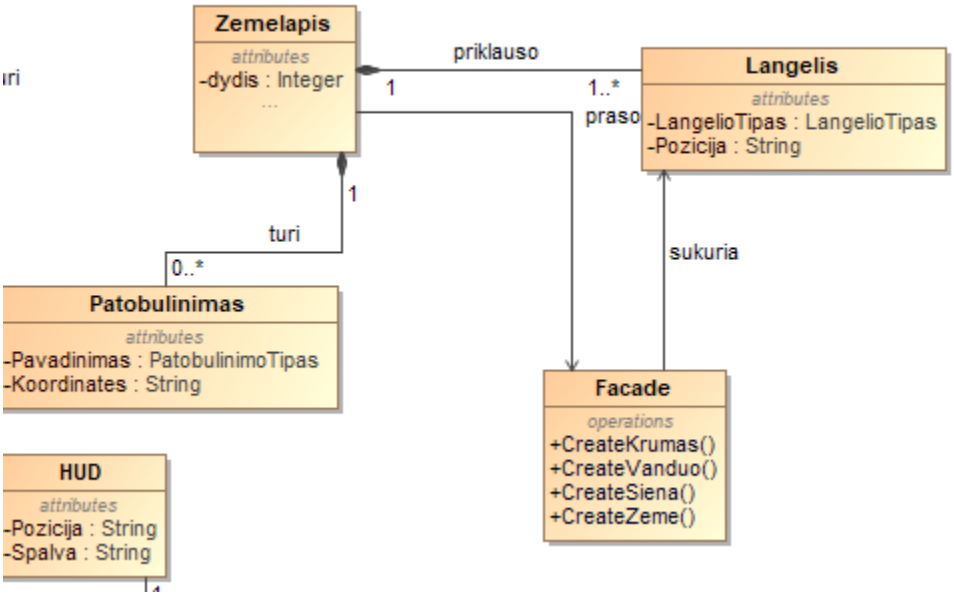
```
namespace TanksRework.Classes.Adapter
{

    public interface IFountain
    {
            int Heal();
    }

}
namespace TanksRework.Classes.Adapter
{
    [JsonObject(MemberSerialization.OptIn)]
    class NormalFountain : IFountain
    {
        [JsonProperty]
        public int HealAmount { get; set; }
        [JsonProperty]
```

```
        public int positionx { get; set; }
        [JsonProperty]
        public int positiony { get; set; }
        public NormalFountain()
        {
            HealAmount = 10;
            positionx = 7;
            positiony = 7;
        }
        public int Heal()
        {
            return HealAmount = 1;
        }
    }
}
namespace TanksRework.Classes.Adapter
{
    class RestoreFountainAdapter : IFountain
    {
        public RestoreFountain fountain;
        public RestoreFountainAdapter(RestoreFountain fountain)
        {
            this.fountain = fountain;
        }
        public int Heal()
        {
            return fountain.Restore();
        }
    }
}
namespace TanksRework.Classes.Adapter
{
    [JsonObject(MemberSerialization.OptIn)]
    class RestoreFountain
    {
        [JsonProperty]
        public int HealAmount { get; set; }
        [JsonProperty]
        public int positionx { get; set; }
        [JsonProperty]
        public int positiony { get; set; }
        public RestoreFountain()
        {
            HealAmount = 10;
            positionx = 7;
            positiony = 7;
        }
        public int Restore()
        {
            return HealAmount;
```

```
            }
        }
    }
```

Facade

Šis projektavimo šablonas yra skirtas tam, kad kurtų skirtingus langelius, kurie skiriasi savo tipais ir pozicijomis. Jis mum palengvino darbą kuriant žemėlapį.



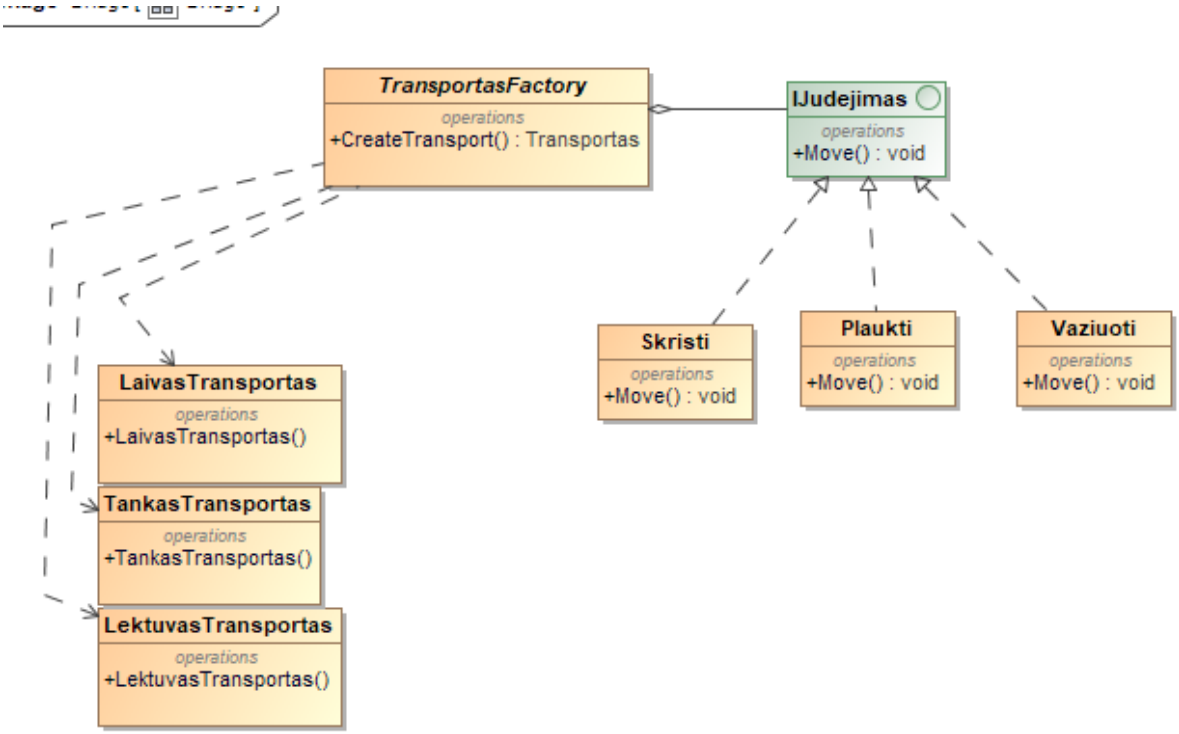
Programinis kodas:

```
namespace TanksRework.Classes
{
    class Facade
    {
        public Langelis CreateKrumas(int x, int y)
        {
            return new Langelis(LangelioTipas.Krumas, x, y);
        }

        public Langelis CreateSiena(int x, int y)
        {
            return new Langelis(LangelioTipas.Siena, x, y);
        }
```

```
        public Langelis CreateVanduo(int x, int y)
        {
            return new Langelis(LangelioTipas.Vanduo, x, y);
        }

        public Langelis CreateZeme(int x, int y)
        {
            return new Langelis(LangelioTipas.Zeme, x, y);
        }
    }
}
```

Bridge

Šis šablonas skirtas sujungti dviem klasėm, kurios abi yra interface arba abstrakčios klasės. Šio šablono dėka sujungėm klases, kurios leido mum kurti skirtingas transporto priemones ir joms priskirti skirtingas judėjimo klases.



Programinis kodas:

```
namespace Classes

{

    [JsonObject(MemberSerialization.OptIn)]

    public abstract class Transportas : Observer.Subject,
Observer.IObserver

    {

        [JsonProperty]

        private string _id { get; set; }

        [JsonProperty]

        private string name { get; set; }

        [JsonProperty]

        private int healthPoints { get; set; }
```

```
        [JsonProperty]
        private int damage { get; set; }
        [JsonProperty]
        public int positionx { get; set; }
        [JsonProperty]
        public int positiony { get; set; }
        [JsonProperty]
        public int type { get; set; }
        [JsonProperty]
        public IJudejimas _strategy { get; set; }



        public Transportas(String nam, int hp, int dmg, int posx, int
posy)
        {
            name = nam;
            healthPoints = hp;
            damage = dmg;
            positionx = posx;
            positiony = posy;
        }
        public Transportas(string id, String nam, int hp, int dmg, int
posx, int posy)
        {
            _id = id;
            name = nam;
            healthPoints = hp;
            damage = dmg;
            positionx = posx;
            positiony = posy;
        }
```

```
        public Transportas()
        {

        }

        public string getId()
        {
            return _id;
        }
        public void setId(string id)
        {
            _id = id;
        }

        void IObserver.atnaujinti(List<Transportas> updPriesai)
        {
            //Atnaujint info (pos = new pos)
            positionx = updPriesai.Find(p => p.getId() ==
_id).positionx;
            positiony = updPriesai.Find(p => p.getId() ==
_id).positiony;
            //if nera tokio id sarase, pridedam prie observeriu?
        }

        public void Move(int x, int y)
        {
            var positions = _strategy.Move(x, y, this.positionx,
this.positiony);
            this.positionx = positions.Item1;
            this.positiony = positions.Item2;
        }
```

```
        }
    }
```